amU Faculté des sciences Aix Marseille Université

Année 2025-26

Domaine Sciences et Technologies
Master informatique
Génie logiciel : TP 3
Codes UE : SINA09EL
Tests : tests boite noire

1 Introduction

Le test en boite noire vise à réaliser des tests sans connaitre le code d'une fonctionnalité ou d'une application. Les tests en boite noire se basent donc sur une spécification de la fonctionnalité qu'on souhaite tester. Les spécifications pouvant être de nature variable, il est difficile d'automatiser la génération des tests en boite noire. On applique plutôt dans ce cas une méthodologie qui nous permettra d'extraire des cas de test qu'on souhaitera le plus exhaustif possible.

Nous allons dans ce TP mettre en pratique une telle méthodologie afin de tester un certain nombre d'implémentations différentes d'une même application. Cette application permet de vérifier la conformité d'un fichier vis-à-vis d'un format de fichiers pour des images.

2 Spécification des formats de fichiers d'images de la librairie Netpbm

2.1 Présentation générale du format

La libraire Netpbm offre plusieurs outils pour la manipulation d'images et définit un certain nombre de formats d'images (le portable pixmap file format (PPM), le portable graymap file format (PGM) et le portable bitmap file format (PBM)), formats dont la spécification est décrite ci-dessous.

Chaque fichier ne comprend la description que d'une seule image. Les fichiers PBM, PGM ou PPM sont composés sur la même base et comporte un entête suivi des données de l'image proprement dite.

Pour ce qui est de l'entête, elle est composée de la manière suivante :

- le nombre magique du format (deux octets) : il indique le type de format (PBM, PGM, ou PPM) et la variante (binaire ou ASCII) ; Le nombre magique est constitué de la lettre `P` suivie d'un chiffre entre 1 et 6 inclus selon le format souhaité :
  - `P1` : PBM en ASCII, extension de fichier `.pbm`
  - `P2` : PGM en ASCII, extension de fichier `.pgm`
  - `P3` : PPM en ASCII, extension de fichier `.ppm`
  - `P4` : PBM en binaire, extension de fichier `.pbm`
  - `P5` : PGM en binaire, extension de fichier `.pgm`
  - `P6` : PPM en binaire, extension de fichier `.ppm`
- l'extension de fichier (`.pbm`, `.pgm` ou `.ppm`) doit correspondre au format indiqué par le nombre magique ;
- un caractère d'espacement (espace `SP`, tabulation `HT`, nouvelle ligne `LF`, `CR`) ;
- la largeur de l'image en texte qui devra correspondre à un entier strictement positif (nombre de pixels, écrit explicitement sous forme d'un nombre en caractères ASCII), suivi d'un caractère d'espacement ;
- la hauteur de l'image en texte qui devra correspondre à un entier strictement positif (idem), suivi d'un caractère d'espacement ;

— enfin pour certains des formats, on trouve toujours codé en ASCII, une information concernant la palette de couleurs utilisées, suivie d'un séparateur (caractère d'espacement) ;
— Toutes les lignes au sein de l'entête commençant par croisillon `#` sont ignorées (lignes de commentaires).

Viennent ensuite les données de l'image comme une succession de valeurs associées à chaque pixel, selon les règles suivantes :

— l'image est codée ligne par ligne en partant du haut,
— chaque ligne de l'image est codée de gauche à droite.

Pour ce qui est des données de l'image proprement dite, nous avons évoqué précédemment pour chaque type deux variantes possibles, la version binaire et la version ASCII. La version binaire est la version "classique" de représentation des données. Ainsi, l'entier 12 en format binaire sera représenté par l'octet 00001100. En revanche, dans le format ASCII, ce même nombre sera représenté comme le caractère `1`, suivi du caractère `2` (codé en ASCII). On note qu'une image au format ASCII sera donc interprétable directement par un humain. Pour les versions ASCII, les caractères d'espacement à l'intérieur des données de l'image serve de séparateurs entre les représentations des valeurs de l'image et aucune ligne ne doit dépasser 70 caractères (règle valable pour les lignes de commentaire mais sans compter le caractère de fin de ligne). Plus précisément, selon les différents formats, on devra respecter les règles suivantes.

2.2 Le format PBM (P1 et P4)

Il s'agit ici d'images en noir et blanc avec la convention que 0 vaut blanc et 1 vaut noir. Il n'y a pas d'information de palette de couleurs dans ce format. Dans la variante binaire, les bits sont regroupés par 8 pour former un octet, les bits excédentaires à la fin d'une ligne sont ignorés. Pour ce qui est de la variante ASCII, les caractères `0` (blanc) et `1` (noir) sont utilisées et les caractères d'espacement à l'intérieur sont ignorés.

2.3 Le format PGM (P2 et P5)

Il s'agit ici d'images en nuance de gris. L'information de la palette de couleurs est la valeur maximale utilisée pour coder les niveaux de gris ; cette valeur doit être inférieure à 65536 (codée en caractères ASCII) et suivie d'un séparateur. Chaque niveau de gris est codé par une valeur entre 0 et cette valeur maximale, proportionnellement à son intensité. Un pixel noir est codé par la valeur 0, un pixel blanc est codé par la valeur maximale.

Pour ce qui est de l'image, dans les données binaires de l'image, chaque pixel est codé par 1 ou 2 octets selon que la valeur maximale soit strictement inférieure (1 octet), ou bien supérieure ou égale à 256 (2 octets). Dans le cas de 2 octets, le premier octet est l'octet de poids fort. Dans le cas de la variante ASCII, chaque pixel est codé par une valeur en caractères ASCII, séparés par des caractères d'espacement.

2.4 Le format PPM (P3 et P6)

Ce format de fichier est utilisé pour des images couleur. Chaque pixel est codé par trois valeurs (rouge, vert et bleu). Comme le format PGM, en plus des caractéristiques de largeur et hauteur, pour la palette, une valeur maximale est utilisée pour coder les niveaux de couleur ; cette valeur doit être inférieure à 65 536 sur chacune des composantes RGB. Ces trois valeurs sont codées en ASCII selon l'ordre RGB et chacune est suivie par un séparateur.

Pour la variante binaire, chaque valeur est codée sur 1 ou 2 octets selon que la valeur maximale soit strictement inférieure ou bien supérieure ou égale à 256. Dans le cas de 2 octets, le premier octet est l'octet de poids fort. Pour la variante ASCII, Chaque pixel est codé en caractères ASCII, précédés et suivis par un caractère d'espacement.

3 Travail à réaliser

3.1 Objectif

L'objectif est donc de mener une campagne de tests pour une application dont le but est de tester la conformité d'un fichier image à la spécification des fichiers de la librairie Netpbm. Ainsi, si le fichier soumis à l'application est au format PBM, PGM ou PPM, alors l'application se terminera correctement. Dans le cas contraire, elle retournera un code d'erreur.

On supposera une utilisation normale de l'application, à savoir un appel avec un unique fichier d'images. Pour concrétiser votre travail, nous vous fournissons 13 implémentations différentes de l'application (de `check00exe` à `check12exe` présents dans un sous-répertoire pour chaque type de système d'exploitation du répertoire `executables` du répertoire `TP 3`) dont au plus une est conforme à la spécification.

— En considérant la spécification ci-dessus, construisez un arbre de décision vous permettant de produire les cas de test les plus exhaustifs possibles.
— En utilisant l'arbre de décision construit ci-dessus, produire les cas de test correspondant.

3.2 Suite de test

Une suite de tests sera constituée de fichiers images conformes ou non aux spécifications des formats PBM, PGM et PPM à placer dans le répertoire `images` du répertoire `TP3` du dépôt git. Chaque fichier possédera un nom de la forme `test{xx}_{r}.{yyy}` avec `{xx}` étant deux caractères formant un nombre compris en `00` et `99`, `{r}` étant soit la lettre `Y` (si le fichier image est conforme et donc le résultat attendu est échec : code 1) ou bien la lettre `N` (fichier non conforme et donc le résultat attendu est succès : code 0) et `{yyy}` aura pour valeur soit `pbm`, soit `pgm`, soit `ppm`. Vous pouvez tester votre suite de tests en utilisant la commande `gradle run`. Cela lancera automatiquement la campagne de tests sur l'ensemble des implémentations fournies et des images que vous avez placées dans le répertoire `images`. Pour chaque implémentation et chaque image, un affichage du type suivant sera produit :

```
check00exe test00_N.ppm, expected: false,   actual: false
```

Cet affichage indique pour l'image le résultat attendu (dépendant du nom de l'image avec `Y` ou `N`) et le résultat effectivement obtenu avec l'exécutable testé. L'affichage sera en vert si le résultat attendu et le résultat obtenu sont identiques, en rouge sinon. À la fin de l'exécution, un résumé sera affiché indiquant la suite passe avec succès ou non.

3.3 Rendu

Vous devrez rendre ce TP le dimanche 7 février 2026 à 23h59 au plus tard via le dépôt git mis à votre disposition. Le rendu devra comprendre :

1. un document (rapport) au **format pdf** à mettre dans le répertoire `CR-TP3` du dépôt git qui contiendra :
   - l'arbre de décision (partiel, car l'arbre complet serait trop complexe) vous ayant servi à générer vos cas de tests
   - une liste d'items comprenant pour chacun d'eux :
     - la description de l'objectif de test.
     - le nom du fichier permettant de tester cet objectif.
     - le résultat attendu (succès ou échec de l'application).
   - une brève explication de la raison pour laquelle ce test permet de couvrir l'objectif de test.
   - une section présentant un bilan de votre campagne de tests (nombre de tests, couverture des objectifs, difficultés rencontrées, etc.)
2. l'ensemble des fichiers images donnant vos cas de test à placer dans le répertoire `images` du répertoire `TP3` du dépôt git.